

ONELoop: for the evaluation of one-loop scalar functions

A. van Hameren

The H. Niewodniczański Institute of Nuclear Physics

Polish Academy of Sciences

Radzikowskiego 152, 31-342 Cracow, Poland

hameren@ifj.edu.pl

July 28, 2010

Abstract

ONELoop is a program to evaluate the one-loop scalar 1-point, 2-point, 3-point and 4-point functions, for all kinematical configurations relevant for collider-physics, and for any non-positive imaginary parts of the internal squared masses. It deals with all UV and IR divergences within dimensional regularization. Furthermore, it provides routines to evaluate these functions using straightforward numerical integration.

⁰This work was partially supported by RTN European Programme, MRTN-CT-2006-035505 (HEPTOOLS, Tools and Precision Calculations for Physics Discoveries at Colliders)

Contents

1	Introduction	2
2	Program description	4
2.1	The 1-point function	4
2.2	2-point functions	5
2.3	The 3-point function	5
2.4	The 4-point function	6
2.5	IR-divergent one-loop functions	7
2.6	Further useful routines	8
2.7	Real routines	8
2.8	Routines using numerical integration	8
2.9	Quadruple precision	9
3	One-loop scalar functions via numerical integration	9
3.1	Contour deformation	11
3.2	IR-divergent scalar integrals	12
4	Summary	13
A	Coefficients for IR-divergent box integrals	16
B	IR-divergent triangle integrals	18

1 Introduction

The experiments at LHC for the study of elementary particles demand precise predictions of signals and potential backgrounds for new physics. This implies that for many processes the hard part has to be evaluated at least at the next-to-leading (NLO) level in perturbation theory, and this includes processes with four or more particles in the final state. There has been a considerable progress in this respect in recent years, resulting in full differential calculations for such processes [1–12]. One of the bottlenecks that had to be resolved in order to achieve this was the calculation of the one-loop amplitude necessary to determine the virtual contribution, which in the mentioned publications involves at least 6-point functions. Developments have reached the stage that by now public programs exist that, given the machinery to calculate tree-level amplitudes, can calculate one-loop amplitudes involving 6-point functions and beyond at a high level of automation [13, 14].

Besides dealing with an increase in combinatorial complexity, moving from tree-level amplitudes to one-loop amplitudes demands the treatment of one-loop integrals. The most successful approach has been to express the one-loop amplitudes as a linear combination of universal one-loop functions, dependent on the particular scattering process under consideration only through

the value of kinematical input parameters. The calculation of the one-loop functions is universal, while the calculation of their coefficients is not. In the “tensor-approach”, these functions are the one-loop tensor functions. They involve a reduction scheme to express the tensor functions in terms of scalar functions, but this is universal. In the “unitarity-approach” the universal functions are the one-loop scalar functions themselves.

As mentioned before, most approaches to calculate one-loop amplitudes demand the calculation of the one-loop scalar functions. These may be infra-red (IR) divergent, and have to be dealt with within a regularization scheme. For NLO calculations it then suffices to have available the scalar 1-point, 2-point, 3-point and 4-point functions. Higher-point scalar functions can be expressed as linear combination of these. The preferred regularization scheme for NLO calculations in QCD is dimensional regularization, and public libraries LOOPTOOLS [17], QCDDLOOP [15] and ONELOOP [16] deliver all necessary scalar function for LHC kinematics and real particle masses. The first two rely on the public library FF for finite scalar functions [18].

Sooner or later, multi-particle NLO calculations will involve unstable particles. The preferred method to perform such calculations in a consistent way is the “complex-mass scheme”, which demands the evaluation of one-loop functions with complex internal masses [19]. LOOPTOOLS provides those, including the most complicated finite 4-point function with 4 complex masses. The latter has been presented in [20], which is the completion and implementation of the work in [21]. Alternative formulas for the 4-point function with complex masses have been given recently in [22].

In this write-up, the extension of ONELOOP to deal with complex masses is presented. Essentially the only missing ingredient was the 4-point function with an arbitrary number of complex masses. It has now been implemented too, using the formulas from [20]. As mentioned before, ONELOOP provides all finite and IR-divergent scalar functions withing dimensional regularization. It does not provide a complete treatment of exceptional phase space points corresponding to Landau singularities. Non-singular exceptional phase space points are taken care of to a certain extend, by expressing formulas in terms of differences of two dilogarithms divided by the difference of their arguments, and using numerically stable implementations of these. However, as argued in [22], these exceptional cases do only rarely appear in practical calculations.

Since the analytic continuation and the implementation of the formulas for the complex routines is rather delicate, it is useful to have the means available to check their correctness numerically. First of all, they have successfully been compared with existing implementations. This includes a private code by A. Denner [23]. As an extra check, they have been compared with results obtained via straightforward numerical integration. The methods applied to achieve this are presented in this write-up, and routines to evaluate all scalar functions involving at least two scales via numerical integration, thereby avoiding the evaluation of any dilogarithm, have been included in the ONELOOP library. The actual integration is performed with the help of the CUBA-library [24] and routines from the QUADPACK library [25].

ONELOOP can be obtained from

<http://helac-phegas.web.cern.ch/>

2 Program description

The main part of the package consists of 10 source files, written in FORTRAN 77 with common extensions, like `enddo` statements and `do while` statements. Furthermore, long names are used, and underscores are used in names. In particular, all names of subroutines, functions and common blocks begin with `avh_olo_`. It is in principle possible to compile the source files and link the object files directly to the main program the routines are supposed to be used in. It may, however, be more practical to create a library with the makefile coming with the package, and link the library instead.

2.1 The 1-point function

The one-loop scalar 1-point function is defined as

$$\text{---} \bigcirc \text{---} \quad m^2 \quad A_0 = h(\mu, \varepsilon) \int \frac{d^{4-2\varepsilon}l}{l^2 - m^2 + i\epsilon}, \quad (1)$$

where the function h of the renormalization scale μ and dimensional regularization parameter $\varepsilon = 2 - D/2$ follows the same convention as in [15], and is given by

$$h(\mu, \varepsilon) = \frac{\Gamma(1-2\varepsilon) \mu^{2\varepsilon}}{\Gamma^2(1-\varepsilon)\Gamma(1+\varepsilon) i\pi^{2-\varepsilon}}. \quad (2)$$

The $i\epsilon$ in the propagator denominators indicates the Feynman prescription for analytic continuation. Squared momenta follow the Bjorken-Drell convention $l^2 = l_0^2 - l_1^2 - l_2^2 - l_3^2$.

The value of μ is by default equal to 1. This default can be changed with

```
subroutine avh_olo_mu_set( mu )
```

The input `mu` shall be real, and has the dimension of energy, not energy squared.

The 1-point function is evaluated with

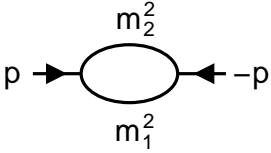
```
subroutine avh_olo_a0c( rslt ,m )
```

The input m shall be complex, and is the squared mass m^2 in Eq.(1). It shall not have a positive imaginary part. If it does, an error message is returned, and the sign of the imaginary part is considered to be the opposite.

The output `rslt` shall be a complex array of shape $(0:2)$, with `rslt(0)` containing the ε^0 coefficient and `rslt(1)` containing the ε^{-1} coefficient of the Laurent expansion in ε of Eq.(1). The value of `rslt(2)` is always equal zero. If `m` is equal zero, then `rslt(0)` and `rslt(1)` are also zero.

2.2 2-point functions

The one-loop scalar 2-point function is defined as



$$B_0 = h(\mu, \varepsilon) \int \frac{d^{4-2\varepsilon} l}{[l^2 - m_1^2 + i\epsilon][(l+p)^2 - m_2^2 + i\epsilon]}, \quad (3)$$

where $h(\mu, \varepsilon)$ is defined in Eq.(2). The 2-point function is evaluated with

```
subroutine avh_olo_b0c( rslt ,p,m1,m2 )
```

The input $p, m1, m2$ shall be complex, and is the square p^2 of the momentum and the squared masses m_1^2 and m_2^2 in Eq.(3) respectively. The imaginary part of p shall be identically zero. If it is not, an error message is returned and the imaginary part is considered to be identically zero. The squared masses shall not have a positive imaginary part. If any of them does, an error message is returned, and the sign of the imaginary part is considered to be the opposite.

The output $rslt$ shall be a complex array of shape $(0:2)$, with $rslt(0)$ containing the ε^0 coefficient and $rslt(1)$ containing the ε^{-1} coefficient of the Laurent expansion in ε of Eq.(3). The value of $rslt(2)$ is always equal zero. If p and $m1$ and $m2$ are equal zero, then $rslt(0)$ and $rslt(1)$ are also zero.

For the 2-point function, also the Passarino-Veltman functions B_1 , B_{00} and B_{11} are available. They are defined following

$$h(\mu, \varepsilon) \int \frac{d^{4-2\varepsilon} l \quad l^\mu}{[l^2 - m_1^2 + i\epsilon][(l+p)^2 - m_2^2 + i\epsilon]} = p^\mu B_1 \quad (4)$$

$$h(\mu, \varepsilon) \int \frac{d^{4-2\varepsilon} l \quad l^\mu l^\nu}{[l^2 - m_1^2 + i\epsilon][(l+p)^2 - m_2^2 + i\epsilon]} = g^{\mu\nu} B_{00} + p^\mu p^\nu B_{11}. \quad (5)$$

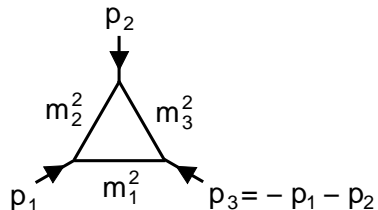
They are evaluated with

```
subroutine avh_olo_b11c( b11,b00,b1,b0 ,p,m1,m2 )
```

The output $b11, b00, b1, b0$ respectively refers to B_{11}, B_{00} and B_1 and B_0 . Each of them shall be a complex array of shape $(0:2)$ again, with the entries containing the coefficients of the Laurent expansion in ε .

2.3 The 3-point function

The one-loop scalar 3-point function is defined as



$$C_0 = h(\mu, \varepsilon) \int \frac{d^{4-2\varepsilon} l}{\text{den}_1(l) \text{den}_2(l) \text{den}_3(l)}, \quad (6)$$

where

$$\begin{aligned} \text{den}_1(l) &= l^2 - m_1^2 + i\epsilon \\ \text{den}_2(l) &= (l + p_1)^2 - m_2^2 + i\epsilon \\ \text{den}_3(l) &= (l + p_1 + p_2)^2 - m_3^2 + i\epsilon, \end{aligned} \quad (7)$$

and $h(\mu, \epsilon)$ is defined in Eq.(2). The 3-point function is evaluated with

subroutine `avh_olo_c0c(rslt , p1, p2, p3 , m1, m2, m3)`

The input p_1, p_2, p_3 shall be complex, and in terms of the momenta in Eq.(7) it is given by

$$p_1^2, p_2^2, p_3^2 = (p_1 + p_2)^2. \quad (8)$$

The imaginary parts of these shall be identically zero. If any of them is not, an error message is returned, and it is considered to be identically zero.

The input m_1, m_2, m_3 shall also be complex, and in terms of the masses in Eq.(7) it is given by

$$m_1^2, m_2^2, m_3^2. \quad (9)$$

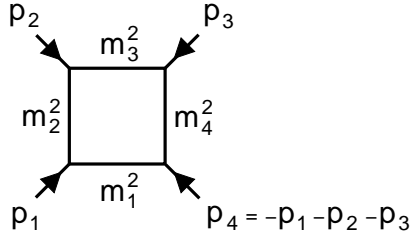
The imaginary parts of these shall not be positive. If any of them is, an error message is returned, and it will be considered to have the opposite sign.

The output `rslt` shall be a complex array of shape $(0:2)$, with `rslt(0)` containing the ϵ^0 coefficient, `rslt(1)` containing the ϵ^{-1} coefficient, and `rslt(2)` containing the ϵ^{-2} coefficient of the Laurent expansion in ϵ of Eq.(6).

By default, the values of `rslt(1)` and `rslt(2)` will be zero if the input does not *exactly* represent a configuration leading to an IR-divergent 3-point function. See Section 2.5 for more information.

2.4 The 4-point function

The one-loop scalar 4-point function is defined as



where

$$\begin{aligned} \text{den}_1(l) &= l^2 - m_1^2 + i\epsilon \\ \text{den}_2(l) &= (l + p_1)^2 - m_2^2 + i\epsilon \\ \text{den}_3(l) &= (l + p_1 + p_2)^2 - m_3^2 + i\epsilon \\ \text{den}_4(l) &= (l + p_1 + p_2 + p_3)^2 - m_4^2 + i\epsilon, \end{aligned} \quad (11)$$

and $h(\mu, \epsilon)$ is defined in Eq.(2). The 4-point function is evaluated with

subroutine avh_olo_d0c(rslt ,p1,p2,p3,p4,p12,p23 ,m1,m2,m3,m4)

The input $p_1, p_2, p_3, p_4, p_{12}, p_{23}$ shall be complex, and in terms of the momenta in Eq.(11) it is given by

$$p_1^2, p_2^2, p_3^2, p_4^2 = (p_1 + p_2 + p_3)^2, (p_1 + p_2)^2, (p_2 + p_3)^2. \quad (12)$$

The imaginary parts of these shall be identically zero. If any of them is not, an error message is returned, and it is considered to be identically zero.

The input m_1, m_2, m_3, m_4 shall also be complex, and in terms of the masses in Eq.(11) it is given by

$$m_1^2, m_2^2, m_3^2, m_4^2. \quad (13)$$

The imaginary parts of these shall not be positive. If any of them is, an error message is returned, and it will be considered to have the opposite sign.

The output `rslt` shall be a complex array of shape $(0:2)$, with `rslt(0)` containing the ε^0 coefficient, `rslt(1)` containing the ε^{-1} coefficient, and `rslt(2)` containing the ε^{-2} coefficient of the Laurent expansion in ε of Eq.(10).

By default, the values of `rslt(1)` and `rslt(2)` will be zero if the input does not *exactly* represent a configuration leading to an IR-divergent 4-point function. See Section 2.5 for more information.

2.5 IR-divergent one-loop functions

ONELOOP deals with all IR-divergent cases for the 3-point and the 4-point function within dimensional regularization. IR-finite cases are defined as those for which the ε^{-1} coefficient and the ε^{-2} coefficient of the Laurent expansion in ε are identically zero. IR-divergent cases are defined as those that are not IR-finite. They are all listed in [15], and can also be found in Appendix A and Appendix B.

By default, ONELOOP only returns IR-divergent cases if the input is exactly referring to a IR-divergent case. This does not mean that the input has to be given in a certain order. Both

```
zero = dcplx(0d0)
call avh_olo_c0c( rslt ,m2,p2,m3 ,zero,m2,m3 )
```

and

```
zero = dcplx(0d0)
call avh_olo_c0c( rslt ,m3,m2,p2 ,m3,zero,m2 )
```

lead to a divergent case. The default does, however, mean that the input has to be numerically exact. For example, if an internal mass being equal to zero represents an IR-divergent case, then by default this divergent case is only returned if this mass is indeed identically zero on input. Also if an external momentum squared being equal to an internal squared mass represents an IR-divergent case, this divergent case is only returned if these numbers are indeed identical on input. *The default can be changed with*

```
subroutine avh_olo_onshell( thrs )
```

The input `thrs` shall be real, and is the threshold below which input is considered to be equal, or equal zero. It carries the unit of energy squared, so if for example the absolute difference of a squared momentum and a squared mass is smaller than `thrs`, they are considered to be equal. Also, if a squared mass or squared momentum is absolutely smaller than `thrs`, it is considered to be equal zero. If the user decides to stick to the default, warnings are returned whenever the input is such that it is close to a divergent case. The user is advised to take these warnings seriously.

The scale associated with the regularization of the IR divergences is the same as the renormalization scale μ of Section 2.1.

2.6 Further useful routines

All messages are by default sent to unit 6. This default can be changed with

```
subroutine avh_olo_unit( unit )
```

The input `unit` shall be integer, and if it is larger than 0, messages will be sent to unit `unit`. If it is not larger than 0, no messages will be printed at all.

ONELoop can be ordered to print all input and output appearing during a run with

```
subroutine avh_olo_printall( unit )
```

The integer input `unit` is the unit to which all this information is sent. Again, if `unit` is not larger than 0, nothing will be printed.

2.7 Real routines

Some users may want to stick to real input. For all routines to evaluate one-loop functions, there are wrappers prepared that ask for strictly real input. For historical reasons, the routine names end on `m` rather than on `r`, and are given by

```
subroutine avh_olo_a0m( rslt ,m )
subroutine avh_olo_b0m( rslt ,p,m1,m2 )
subroutine avh_olo_b11m( b11,b00,b1,b0 ,p,m1,m2 )
subroutine avh_olo_c0m( rslt ,p1,p2,p3 ,m1,m2,m3 )
subroutine avh_olo_d0m( rslt ,p1,p2,p3,p4,p12,p23 ,m1,m2,m3,m4 )
```

2.8 Routines using numerical integration

As mentioned in the introduction, with the ONELoop package also come routines to evaluate one-loop functions using numerical integration. These have been included as means of a cross-check. The main purpose is to check the correctness of analytic continuations, and the emphasis

has been put on straightforwardness rather than sophistication in their implementation. In particular the routines for the 4-point function can be slow, and will give only the first few decimals correctly. Since errors in analytic continuation lead to rather drastic differences, not just in the last few decimals, this is in general sufficient.

The routines come as an independent library, which from user point of view essentially is a copy of the main ONELOOP library. It has all routines, except those for the 2-point Passarino-Veltman functions, with the only difference that the routine names start with `avh_oni` instead of `avh_olo`. It has to be compiled separately, and needs access to the `cure` routine of the CUBA library [24]. It provides two extra routines, namely

```
subroutine avh_oni_maxeval_set( maxeval )
subroutine avh_oni_unitcuba_set( unit )
```

The integer input `maxeval` of the first routine changes the maximal number of integrand evaluations `cure` is ordered to perform, which by default is 10^6 . The integer input `unit` of the second routine changes the unit to which messages related to the numerical integration are sent, which is 6 by default. If this number is not larger than 0, no messages will be printed.

2.9 Quadruple precision

Some compilers provide the option to compile at a higher precision level, such that for example all 16-byte complex variables become 32-byte variables. All series expansions inside ONELOOP have been prepared to also deal with this precision level. During its first execution, ONELOOP will determine the precision level at which it has been compiled, and from then on use the relevant expansions.

3 One-loop scalar functions via numerical integration

In this section, integral representations for the one-loop scalar functions suitable for numerical integration are derived. Since the aim is to have an alternative numerical implementation of the scalar functions in order to check the correctness of the analytic continuations in the “hard-wired” implementation, the main guideline in the derivation is that it should be straightforward. Methods to smoothen the integrand in order to speed up numerical integration, as for example used in [26,27], are not applied. Other approaches to numerical integration of loop integrals can be found in [28–30, 32–34].

The main problem in the numerical integration of loop integrals is the existence of singularities in the integrand. These can be categorized into two groups: those that can be dealt with via deformation of the integration contour, and those that cannot. The former will be dealt with... via contour deformation. Among the latter are the so-called mass-singularities, or infra-red (IR) singularities. They cause the integral to be divergent, and are typically inherent to the particular scattering process under consideration. They ask for a regularization scheme, and are dealt with

within dimensional regularization in this section. Other singularities that cannot be dealt with via contour deformation typically appear in isolated points in phase space. A complete treatment of those, like for example in [27], is beyond the scope of this work.

In this section, we write the one-loop n -point scalar integral within dimensional regularization in the following convention

$$I_n = \frac{(-)^n}{\Gamma(n-2+\epsilon) i\pi^{2-\epsilon}} \int \frac{d^{4-2\epsilon} l}{\prod_{j=1}^n [(l+q_j)^2 - m_j^2 + i\epsilon]} . \quad (14)$$

The denominator momenta q_j are related to the external momenta p_j via $q_1 = 0$ and

$$q_j = \sum_{k=1}^{j-1} p_k . \quad (15)$$

The reason for the choice of the factor in front of the integral is that it leads to a simple expression for the scalar function in terms of the so-called Feynman parameters, namely

$$I_n = \int_{\Omega_n} d^n x \frac{\delta(1 - \sum_{j=1}^n x_j)}{(\vec{x} \cdot S \vec{x} - i\epsilon)^{n-2+\epsilon}} \quad (16)$$

where S is the linear operator represented by the symmetric $n \times n$ matrix with elements

$$S_{j,k} = \frac{m_j^2 + m_k^2 - (q_j - q_k)^2}{2} . \quad (17)$$

The integration region is such that all variables x_j run from 0 to ∞ ,

$$\Omega_n = [0, \infty)^n , \quad (18)$$

however the Dirac delta-function in Eq.(16) confines the integration region eventually to the n -simplex. One could proceed by direct elimination of one of the integration variables with the help of the delta-function. We shall take a slightly different strategy, and scale the variables x_1 to x_{n-1} by x_n , which we call y , to find

$$I_n = \int_0^\infty dy y^{n-1} \int_{\Omega_{n-1}} d^{n-1} x \frac{\delta(1 - y(1 + \sum_{j=1}^{n-1} x_j))}{y^{2n-4+2\epsilon} (\vec{x} \cdot A \vec{x} + 2\vec{b} \cdot \vec{x} + c)^{n-2+\epsilon}} . \quad (19)$$

Here, we denote

$$A = \begin{pmatrix} S_{1,1} & S_{1,2} & \cdots & S_{1,n-1} \\ S_{2,1} & S_{2,2} & \cdots & S_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ S_{n-1,1} & S_{n-1,2} & \cdots & S_{n-1,n-1} \end{pmatrix} , \quad \vec{b} = \begin{pmatrix} S_{n,1} \\ S_{n,2} \\ \vdots \\ S_{n,n-1} \end{pmatrix} , \quad c = S_{nn} - i\epsilon . \quad (20)$$

We used the fact that we are allowed to replace ϵ by $y^2\epsilon$ since y^2 is positive. Elimination of the y -integral now gives

$$I_n = \int_{\Omega_{n-1}} d^{n-1} x \frac{(1 + \sum_{j=1}^{n-1} x_j)^{n-4+2\epsilon}}{(\vec{x} \cdot A \vec{x} + 2\vec{b} \cdot \vec{x} + c)^{n-2+\epsilon}} . \quad (21)$$

The integration region is not finite anymore. One advantage of expressing the one-loop integral as Eq.(21) is that one does not have to bother about the end-point at infinity if one wants to explicitly deform the integration contour in the complex plane in order to avoid poles as prescribed by i.e. It does not matter towards which direction in the complex plane infinity is approached, since the integrand does not have a singularity at infinity. Another advantage is that a proliferation of terms is prevented when one wants to apply techniques involving integration by parts, since integrands will be 0 at infinity.

3.1 Contour deformation

The iε-prescription dictates that the contour deformation has to be such that the imaginary part of the polynomial in the denominator of the integrand in Eq.(21) is negative when the real part vanishes. Furthermore, the deformation has to be such that the end-points at $x_j = 0$ (the begin-points) stay fixed. IR-singularities, which cannot be dealt with by contour deformation, may show up as vanishing real parts at $x_j = 0$.

Since we want to allow for squared masses with non-zero, but negative, imaginary parts, the matrix A and the vector \vec{b} may have imaginary parts. We re-define our notation, and write

$$A \leftarrow A - i\Gamma \quad , \quad \vec{b} \leftarrow \vec{b} - i\vec{\gamma} \quad , \quad (22)$$

now with A and \vec{b} real. The matrix Γ and the vector $\vec{\gamma}$ have only non-negative components. Our deformation is very similar to the one in [32]. We introduce a strictly increasing function τ such that $\tau(0) = 0$ and $\tau(\infty)$ is finite, for example $\tau(x) = \tau_\infty x / (1 + x)$ with $\tau_\infty > 0$. The deformation is given by

$$\vec{x} \mapsto \vec{z}(\vec{x}) = \vec{x} - i\vec{y}(\vec{x}) \quad , \quad (23)$$

with

$$y_j(\vec{x}) = \tau(x_j)(A\vec{x} + \vec{b})_j \quad , \quad (24)$$

and where $(A\vec{x} + \vec{b})_j$ denotes the j -th component of the vector $A\vec{x} + \vec{b}$. The Jacobian matrix coming with the deformation is given by

$$\frac{\partial z_j}{\partial x_k} = \left[1 - i\tau'(x_j)(A\vec{x} + \vec{b})_j \right] \delta_{j,k} - i\tau(x_j)A_{j,k} \quad . \quad (25)$$

The imaginary part of the polynomial in the denominator of the integrand in Eq.(21) satisfies

$$-\text{Im} \left\{ \vec{z} \cdot (A - i\Gamma)\vec{z} + 2(\vec{b} - i\vec{\gamma}) \cdot \vec{z} + c \right\} = 2\vec{y} \cdot (A\vec{x} + \vec{b}) - \vec{y} \cdot \Gamma \vec{y} + \vec{x} \cdot \Gamma \vec{x} + 2\vec{\gamma} \cdot \vec{x} - \text{Im}[c] \quad . \quad (26)$$

The last three terms are safe, in the sense that they never become negative. The first term never becomes negative if \vec{y} is chosen as in Eq.(24) (notice that the function τ acts as a deformation of the inner product, keeping it however positive definite). We have to worry a bit about the second term. The matrix Γ has elements

$$\Gamma_{j,k} = \eta_j + \eta_k \quad , \quad \eta_j = -\frac{1}{2}\text{Im}\{m_j^2\} \geq 0 \quad , \quad (27)$$

so

$$\vec{y} \cdot \Gamma \vec{y} = 2 \left(\sum_{j=1}^{n-1} y_j \right) \left(\sum_{j=1}^{n-1} \eta_j y_j \right). \quad (28)$$

Applying the Schwartz inequality to both sums, we find

$$|\vec{y} \cdot \Gamma \vec{y}| \leq 2\sqrt{n-1} \|\vec{\eta}\| \|\vec{y}\|^2, \quad \|\vec{\eta}\|^2 = \sum_{j=1}^{n-1} \eta_j^2. \quad (29)$$

Now we have for the first two terms on the r.h.s. of Eq.(26)

$$2\vec{y} \cdot (A\vec{x} + \vec{b}) - \vec{y} \cdot \Gamma \vec{y} \geq 2 \sum_{j=1}^{n-1} \left(\frac{y_j^2}{\tau(x_j)} - \sqrt{n-1} \|\vec{\eta}\| y_j^2 \right), \quad (30)$$

and we conclude that we are safe if we choose τ such that

$$\tau(\infty) \leq \frac{1}{\sqrt{n-1} \|\vec{\eta}\|} = \left((n-1) \sum_{j=1}^{n-1} \text{Im}\{m_j^2\}^2 \right)^{-1/2}. \quad (31)$$

Many methods for numerical integration ask for integrands living on the unit hypercube. For our application, this can conveniently be achieved by preceding the contour deformation with the mapping

$$[0, 1] \mapsto [0, \infty) \quad , \quad \rho_j \mapsto x_j = 1/\rho_j - 1, \quad (32)$$

leading to a Jacobian weight factor $\rho_j^{-2} = (1 + x_j)^2$ for each integration variable ρ_j in the integrand.

3.2 IR-divergent scalar integrals

For IR-finite integrals, the integrand in Eq.(21) can be expanded in ε and integrated term by term. In particular, ε can be put to 0 to calculate the ε^0 -term. For IR-divergent integrals, this is not directly possible. In this work, we are only interested in scalar integrals up to $n = 4$, *i.e.* box integrals and triangle integrals. IR-divergent bubble integrals vanish within dimensional regularization.

Regarding the box integrals, we use the fact that the IR-divergent structure of one-loop integrals with more than 3 external lines can be expressed in terms of triangle integrals. This is completely worked out in [35]. Formulated for our application, that work explains how to determine the coefficients c_j for every IR-divergent box integral such that it can be written as

$$I_4 = \sum_{j=1}^4 c_j I_3(j) - I_4^{\text{fin}} \quad (33)$$

where $I_3(j)$ is the triangle integral obtained from the box integral by removing propagator denominator j , and where I_4^{fin} is IR-finite.¹ Putting the terms under one integral and with a common denominator, we can write

$$I_4^{\text{fin}} = \sum_{j=1}^4 c_j I_3(j) - I_4 = \frac{1}{\Gamma(2+\varepsilon) i\pi^{2-\varepsilon}} \int d^{4-2\varepsilon} l \frac{\alpha l^2 + 2v^\mu l_\mu + \gamma}{\prod_{j=1}^4 [(l+q_j)^2 - m_j^2 + i\epsilon]}, \quad (34)$$

with

$$\alpha = \sum_{j=1}^4 c_j, \quad v^\mu = \sum_{j=1}^4 c_j q_j^\mu, \quad \gamma = \sum_{j=1}^4 c_j (q_j^2 - m_j^2) - 1. \quad (35)$$

The integral in Eq.(34) is IR-finite, and can be integrated numerically. The appearance of the term γ in the numerator seems counter-intuitive in this respect, and in fact the coefficients c_j appear to be such that it vanishes for every IR-divergent box integral. In terms of Feynman parameters, I_4^{fin} is given by

$$I_4^{\text{fin}} = \int_{\Omega_4} d^4 x \delta\left(1 - \sum_{j=1}^4 x_j\right) \frac{\alpha(\vec{x} \cdot Q \vec{x} - \frac{2}{1+\varepsilon} \vec{x} \cdot S \vec{x}) - \vec{\beta} \cdot \vec{x} + \gamma}{(\vec{x} \cdot S \vec{x} - i\epsilon)^{2+\varepsilon}}, \quad (36)$$

where the matrix Q and the vector $\vec{\beta}$ have components

$$Q_{j,k} = q_{j,\mu} q_k^\mu, \quad \beta_j = 2v_\mu q_j^\mu. \quad (37)$$

To summarize, the IR-divergent box integrals are calculated following Eq.(33), where I_4^{fin} is finite and can be integrated numerically directly. The coefficients c_j for the triangle integrals can be determined as explained in [35], and are explicitly given in [22]. They can also be found in Appendix A, together with the coefficients α and β_j . For the IR-divergent triangle integrals themselves integral representations are derived in Appendix B which can be performed numerically.

4 Summary

The program ONELOOP for the evaluation of one-loop scalar functions up to and including 4-point functions was presented. It deals with all kinematical configurations relevant for collider-physics, and any non-positive imaginary parts for the internal squared masses, and it deals with all UV and IR divergences within dimensional regularization. Furthermore, it provides routines to evaluate all these functions using straightforward numerical integration, and the methods applied to achieve this were presented. The program can be obtained from

<http://helac-phegas.web.cern.ch/>

¹Within the convention of Eq.(14), $I_3(j)$ actually is a triangle integral divided by $-(1+\varepsilon)$.

References

- [1] A. Bredenstein, A. Denner, S. Dittmaier and S. Pozzorini, JHEP **0808** (2008) 108 [arXiv:0807.1248 [hep-ph]].
- [2] A. Bredenstein, A. Denner, S. Dittmaier and S. Pozzorini, Phys. Rev. Lett. **103** (2009) 012002 [arXiv:0905.0110 [hep-ph]].
- [3] A. Bredenstein, A. Denner, S. Dittmaier and S. Pozzorini, JHEP **1003** (2010) 021 [arXiv:1001.4006 [hep-ph]].
- [4] R. K. Ellis, K. Melnikov and G. Zanderighi, JHEP **0904** (2009) 077 [arXiv:0901.4101 [hep-ph]].
- [5] R. Keith Ellis, K. Melnikov and G. Zanderighi, Phys. Rev. D **80** (2009) 094002 [arXiv:0906.1445 [hep-ph]].
- [6] K. Melnikov and G. Zanderighi, Phys. Rev. D **81** (2010) 074025 [arXiv:0910.3671 [hep-ph]].
- [7] C. F. Berger *et al.*, Phys. Rev. Lett. **102** (2009) 222001 [arXiv:0902.2760 [hep-ph]].
- [8] C. F. Berger *et al.*, Phys. Rev. D **80** (2009) 074036 [arXiv:0907.1984 [hep-ph]].
- [9] C. F. Berger *et al.*, arXiv:1004.1659 [hep-ph].
- [10] G. Bevilacqua, M. Czakon, C. G. Papadopoulos, R. Pittau and M. Worek, JHEP **0909** (2009) 109 [arXiv:0907.4723 [hep-ph]].
- [11] G. Bevilacqua, M. Czakon, C. G. Papadopoulos and M. Worek, Phys. Rev. Lett. **104** (2010) 162002 [arXiv:1002.4009 [hep-ph]].
- [12] T. Binoth, N. Greiner, A. Guffanti, J. Reuter, J. P. Guillet and T. Reiter, Phys. Lett. B **685** (2010) 293 [arXiv:0910.4379 [hep-ph]].
- [13] G. Ossola, C. G. Papadopoulos and R. Pittau, JHEP **0803** (2008) 042 [arXiv:0711.3596 [hep-ph]].
- [14] P. Mastrolia, G. Ossola, T. Reiter and F. Tramontano, arXiv:1006.0710 [hep-ph].
- [15] R. K. Ellis and G. Zanderighi, JHEP **0802** (2008) 002 [arXiv:0712.1851 [hep-ph]].
- [16] A. van Hameren, C. G. Papadopoulos and R. Pittau, JHEP **0909**, 106 (2009) [arXiv:0903.4665 [hep-ph]].
- [17] T. Hahn and M. Perez-Victoria, Comput. Phys. Commun. **118**, 153 (1999) [arXiv:hep-ph/9807565].

- [18] G. J. van Oldenborgh and J. A. M. Vermaseren, Z. Phys. C **46** (1990) 425.
- [19] A. Denner, S. Dittmaier, M. Roth and L. H. Wieders, Nucl. Phys. B **724** (2005) 247 [arXiv:hep-ph/0505042].
- [20] D. T. Nhung and L. D. Ninh, Comput. Phys. Commun. **180** (2009) 2258 [arXiv:0902.0325 [hep-ph]].
- [21] G. 't Hooft and M. J. G. Veltman, Nucl. Phys. B **153** (1979) 365.
- [22] A. Denner and S. Dittmaier, arXiv:1005.2076 [hep-ph].
- [23] W. Beenakker and A. Denner, Nucl. Phys. B **338** (1990) 349. A. Denner, U. Nierste and R. Scharf, Nucl. Phys. B **367**, 637 (1991). A. Denner and S. Dittmaier, Nucl. Phys. B **658** (2003) 175 [arXiv:hep-ph/0212259]. A. Denner and S. Dittmaier, Nucl. Phys. B **734** (2006) 62 [arXiv:hep-ph/0509141].
- [24] T. Hahn, Comput. Phys. Commun. **168** (2005) 78 [arXiv:hep-ph/0404043].
- [25] R. Piessens, E. de Doncker, C. Überhuber, D. Kahaner, Springer-Verlag, 1983.
- [26] G. Passarino, Nucl. Phys. B **619** (2001) 257 [arXiv:hep-ph/0108252].
- [27] A. Ferroglia, M. Passera, G. Passarino and S. Uccirati, Nucl. Phys. B **650**, 162 (2003) [arXiv:hep-ph/0209219].
- [28] Z. Nagy and D. E. Soper, JHEP **0309** (2003) 055 [arXiv:hep-ph/0308127].
- [29] Z. Nagy and D. E. Soper, Phys. Rev. D **74** (2006) 093006 [arXiv:hep-ph/0610028].
- [30] W. Gong, Z. Nagy and D. E. Soper, Phys. Rev. D **79** (2009) 033005 [arXiv:0812.3686 [hep-ph]].
- [31] E. de Doncker, Y. Shimizu, J. Fujimoto and F. Yuasa, Comput. Phys. Commun. **159** (2004) 145.
- [32] T. Binoth, J. P. Guillet, G. Heinrich, E. Pilon and C. Schubert, JHEP **0510** (2005) 015 [arXiv:hep-ph/0504267].
- [33] Y. Kurihara and T. Kaneko, Comput. Phys. Commun. **174** (2006) 530 [arXiv:hep-ph/0503003]. [34]
- [34] C. Anastasiou, S. Beerli and A. Daleo, JHEP **0705** (2007) 071 [arXiv:hep-ph/0703282].
- [35] S. Dittmaier, Nucl. Phys. B **675** (2003) 447 [arXiv:hep-ph/0308246].

A Coefficients for IR-divergent box integrals

In this appendix, the coefficients c_j for Eq.(33) and α, β_j for Eq.(36) are given. The coefficient γ in the latter equation vanishes for all cases. The coefficients are expressed in terms of the external momenta defined as in Eq.(10), and we denote

$$s_{12} = (p_1 + p_2)^2, \quad s_{23} = (p_2 + p_3)^2. \quad (38)$$

For completeness, we also give the matrix S in Eq.(36)

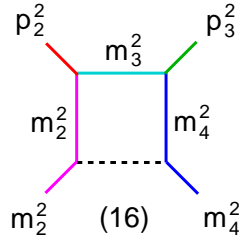
$$\frac{1}{2} \begin{pmatrix} 2m_1^2 & m_1^2 + m_2^2 - p_1^2 & m_1^2 + m_3^2 - s_{12} & m_1^2 + m_4^2 - p_4^2 \\ m_1^2 + m_2^2 - p_1^2 & 2m_2^2 & m_2^2 + m_3^2 - p_2^2 & m_2^2 + m_4^2 - s_{23} \\ m_1^2 + m_3^2 - s_{12} & m_2^2 + m_3^2 - p_2^2 & 2m_3^2 & m_3^2 + m_4^2 - p_3^2 \\ m_1^2 + m_4^2 - p_4^2 & m_2^2 + m_4^2 - s_{23} & m_3^2 + m_4^2 - p_3^2 & 2m_4^2 \end{pmatrix}, \quad (39)$$

and the matrix Q in Eq.(36)

$$\frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2p_1^2 & s_{12} + p_1^2 - p_2^2 & p_1^2 + p_4^2 - s_{23} \\ 0 & s_{12} + p_1^2 - p_2^2 & 2s_{12} & s_{12} + p_4^2 - p_3^2 \\ 0 & p_1^2 + p_4^2 - s_{23} & s_{12} + p_4^2 - p_3^2 & 2p_4^2 \end{pmatrix}. \quad (40)$$

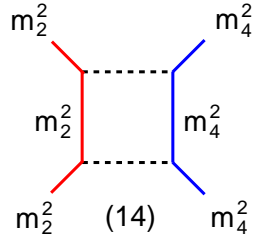
The numbering of the IR-divergent box integrals refers to the numbering in [15]. Some of the separately labeled box integrals in that publication are just particular cases of other box integrals with one or more propagator masses equal to zero. Only the general cases are drawn below. The graphs follow the same conventions as in [15]. Dashed lines correspond to vanishing invariants. Explicitly mentioned invariants are non-zero and not equal to other explicitly mentioned invariants. External lines with the same color as internal lines, and with the same label in the form of a squared mass, indicate that the squared external momentum is equal to the squared internal mass.

Box (16) and (15): one soft and no collinear singularity. m_3^2 may vanish.



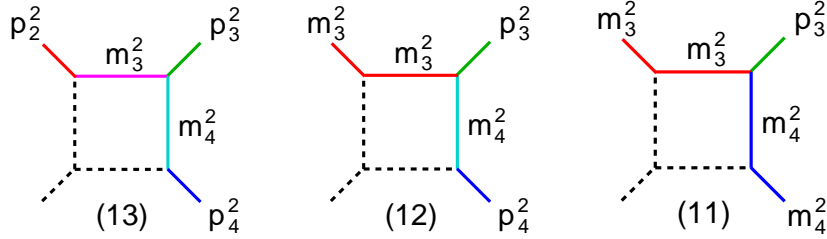
$$\begin{aligned} \vec{c} &= \frac{1}{s_{12} - m_3^2} (0, 0, 1, 0) \\ \alpha &= \frac{1}{s_{12} - m_3^2} \\ \vec{\beta} &= \frac{1}{s_{12} - m_3^2} (0, s_{12} - p_2^2 + m_2^2, 2s_{12}, s_{12} - s_3 + m_4^2). \end{aligned} \quad (41)$$

Box (14): two soft and no collinear singularities.



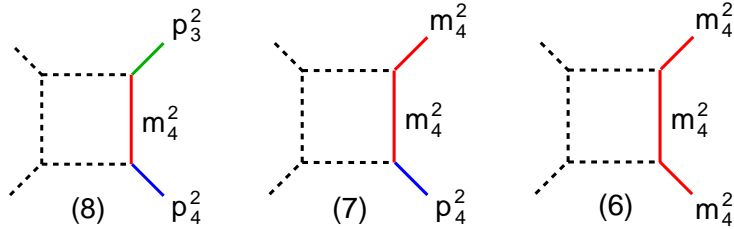
$$\vec{c} = \frac{1}{s_{12}} (1, 0, 1, 0) \quad \alpha = \frac{2}{s_{12}} \quad \vec{\beta} = (0, 1, 2, 1) . \quad (42)$$

Box (13), (12), (11), (10), (9) and (5): one collinear singularity, and no, one, or two soft singularities. m_3^2 and m_4^2 may vanish in box (13), and m_4^2 may vanish in box (12).



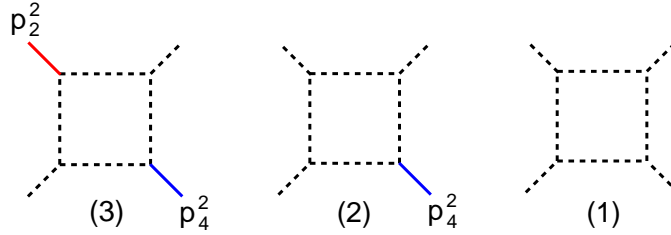
$$\begin{aligned} \Delta &= (s_{12} - m_3^2)(s_{23} - m_4^2) - (p_2^2 - m_3^2)(p_4^2 - m_4^2) \\ \vec{c} &= \frac{1}{\Delta} (0, 0, s_{23} - p_4^2, s_{12} - p_2^2) \\ \alpha &= \frac{s_{12} + s_{23} - p_2^2 - p_4^2}{\Delta} \\ \vec{\beta} &= \frac{1}{\Delta} (0, 0, s_{12}(s_{12} + 2s_{23} - p_4^2 - p_3^2 - p_2^2) + p_2^2(p_3^2 - p_4^2) \\ &\quad , p_4^2(s_{12} + s_{23} + p_3^2 - p_4^2 - 2p_2^2) + s_{23}(s_{12} - p_3^2)) . \end{aligned} \quad (43)$$

Box (8), (7), (6) and (4): two collinear and at least one soft singularity. m_4^2 may vanish in box (8).



$$\begin{aligned}
\vec{c} &= \frac{1}{s_{12}(s_{23} - m_4^2)} (s_{23} - p_3^2, 0, s_{23} - p_4^2, s_{12}) \\
\alpha &= \frac{s_{12} + 2s_{23} - p_4^2 - p_3^2}{s_{12}(s_{23} - m_4^2)} \\
\vec{\beta} &= \frac{1}{s_{12}(s_{23} - m_4^2)} (0, 0, s_{12}(s_{12} + 2s_{23} - p_4^2 - p_3^2) \\
&\quad , p_4^2(s_{12} + s_{23} + p_3^2 - p_4^2) + s_{23}(s_{12} - p_3^2)) .
\end{aligned} \tag{44}$$

Box (3), (2) and (1): more than two collinear, or two collinear and no soft singularities.



$$\begin{aligned}
\vec{c} &= \frac{1}{s_{12}s_{23} - p_2^2 p_4^2} (s_{23} - p_2^2, s_{12} - p_4^2, s_{23} - p_4^2, s_{12} - p_2^2) \\
\alpha &= 2 \frac{s_{12} + s_{23} - p_2^2 - p_4^2}{s_{12}s_{23} - p_2^2 p_4^2} \\
\vec{\beta} &= \alpha (0, 0, s_{12}, p_4^2) .
\end{aligned} \tag{45}$$

B IR-divergent triangle integrals

In this appendix the integral representations for the IR-divergent triangles are given, suitable for numerical integration. The aim is to arrive at those with as few algebraic manipulations as possible, starting from the representation of Eq.(21). This will be done only for triangles involving at least two mass scales (besides the extra scale μ due to the regularization), since the analytic formulas for the triangles involving only one mass scale are rather simple. The considered triangles are depicted in Fig.1, following the labeling as in [15]. The general triangle modified Caley matrix is

$$\frac{1}{2} \begin{pmatrix} 2m_1^2 & m_1^2 + m_2^2 - p_1^2 & m_1^2 + m_3^2 - p_3^2 \\ m_1^2 + m_2^2 - p_1^2 & 2m_2^2 & m_2^2 + m_3^2 - p_2^2 \\ m_1^2 + m_3^2 - p_3^2 & m_2^2 + m_3^2 - p_2^2 & 2m_3^2 \end{pmatrix} . \tag{46}$$

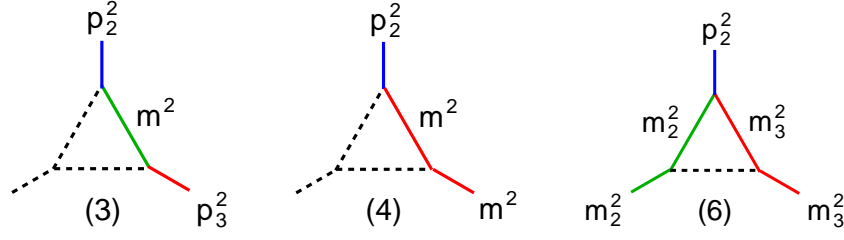


Figure 1: IR-divergent scalar triangles with at least two mass scales, labeled as in [15]. Triangle (2) in that publication is a particular case of triangle (3) with $m^2 = 0$.

Triangle (6)

Triangle (6) has modified Caley matrix

$$\frac{1}{2} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2m_2^2 & m_2^2 + m_3^2 - p_2^2 \\ 0 & m_2^2 + m_3^2 - p_2^2 & 2m_3^2 \end{pmatrix}. \quad (47)$$

The integral representation Eq.(21) of this triangle is given by

$$I_3(6) = \int_0^\infty dx \int_0^\infty dy \frac{(1+x+y)^{-1+2\epsilon}}{(m_2^2 y^2 + (m_2^2 + m_3^2 - p_2^2)y + m_3^2)^{1+\epsilon}} \quad (48)$$

The x -integral can be performed directly, leading to

$$I_3(6) = \frac{-1}{2\epsilon} \int_0^\infty dy \frac{(1+y)^{2\epsilon}}{(m_2^2 y^2 + (m_2^2 + m_3^2 - p_2^2)y + m_3^2)^{1+\epsilon}}. \quad (49)$$

The remaining integral is convergent for any value of ϵ . The integrand can be expanded into a Taylor series in ϵ , which can be integrated numerically term by term.

Triangle (3)

Triangle (3) has modified Caley matrix

$$\frac{1}{2} \begin{pmatrix} 0 & 0 & m^2 - p_3^2 \\ 0 & 0 & m^2 - p_2^2 \\ m^2 - p_3^2 & m^2 - p_2^2 & 2m^2 \end{pmatrix}. \quad (50)$$

We choose to eliminate row and column 1 instead of 3 to obtain the integral representation Eq.(21), leading to

$$I_3(3) = \int_0^\infty dx \int_0^\infty dy \frac{(1+x+y)^{-1+2\epsilon}}{(m^2 x^2 + (m^2 - p_2^2)xy + (m^2 - p_3^2)x)^{1+\epsilon}}. \quad (51)$$

Whereas the integral for triangle (6) was divergent due to the large- x behavior, this integral is divergent due to the small- x behavior. Notice that a factor $x^{1+\varepsilon}$ can be taken out of the denominator. Using this fact, we apply subtraction to write

$$I_3(3) = \int_0^\infty dx \int_0^\infty dy \frac{(1+x+y)^{-1+2\varepsilon}}{x^{1+\varepsilon}} \left[\frac{1}{(m^2x + (m^2 - p_2^2)y + m^2 - p_3^2)^{1+\varepsilon}} - \frac{1}{((m^2 - p_2^2)y + m^2 - p_3^2)^{1+\varepsilon}} \right] + \frac{\pi^{1/2} 4^\varepsilon \Gamma(-\varepsilon)}{\Gamma(\frac{1}{2} - \varepsilon)} \int_0^\infty dy \frac{(1+y)^{-1+\varepsilon}}{((m^2 - p_2^2)y + m^2 - p_3^2)^{1+\varepsilon}}, \quad (52)$$

where we used the identity

$$\int_0^\infty dx \frac{(1+x+y)^{-1+2\varepsilon}}{x^{1+\varepsilon}} = \frac{\pi^{1/2} 4^\varepsilon \Gamma(-\varepsilon)}{\Gamma(\frac{1}{2} - \varepsilon)} (1+y)^{-1+\varepsilon}. \quad (53)$$

Both integrals in Eq.(52) are now finite. To obtain the ε^{-1} -term and the ε^0 -term, the integrand of the second integral may be expanded to order ε , and ε may be put to zero in the first integral. Notice that the first, 2-dimensional, integral vanishes for $m^2 = 0$, *i.e.* for triangle (2) in [15].

Triangle (4)

The integral representation for triangle (4) becomes

$$I_3(4) = \int_0^\infty dx \int_0^\infty dy \frac{(1+x+y)^{-1+2\varepsilon}}{(m^2x^2 + (m^2 - p_2^2)xy)^{1+\varepsilon}}. \quad (54)$$

Rescaling the variable $y \leftarrow xy$, we get

$$I_3(4) = \int_0^\infty dx \int_0^\infty dy \frac{(1+x+xy)^{-1+2\varepsilon}}{x^{1+2\varepsilon}(m^2 + (m^2 - p_2^2)y)^{1+\varepsilon}}, \quad (55)$$

and performing the x -integral, we find

$$I_3(4) = \frac{-1}{2\varepsilon} \int_0^\infty dy \frac{(1+y)^{2\varepsilon}}{(m^2 + (m^2 - p_2^2)y)^{1+\varepsilon}}. \quad (56)$$

Substituting $y \leftarrow 1/y$ we get

$$I_3(4) = \frac{-1}{2\varepsilon} \int_0^\infty dy \frac{(1+y)^{2\varepsilon}}{y^{1+\varepsilon}} \frac{1}{(m^2y + m^2 - p_2^2)^{1+\varepsilon}}, \quad (57)$$

which can be dealt with using subtraction

$$I_3(4) = \frac{-1}{2\varepsilon} \left\{ \int_0^\infty dy \frac{(1+y)^{2\varepsilon}}{y^{1+\varepsilon}} \left[\frac{1}{(m^2y + m^2 - p_2^2)^{1+\varepsilon}} - \frac{1}{(1+y)(m^2 - p_2^2)^{1+\varepsilon}} \right] + \frac{\pi^{1/2} 4^\varepsilon \Gamma(-\varepsilon)}{\Gamma(\frac{1}{2} - \varepsilon)} \frac{1}{(m^2 - p_2^2)^{1+\varepsilon}} \right\}. \quad (58)$$

The remaining integral is finite, and can be expanded in ε on the integrand level.